in a signal processing transform, one or more numbers may each appear in several different products.

DESCRIPTION - FAST FOURIER TRANSFORM TECHNIQUES

There are many techniques for computation of the DFT and inverse DFT which reduce the number of multiplication operations required. These techniques are known as fast Fourier transform, or FFT, techniques. As mentioned above, computation of an N-point DFT or inverse DFT directly from equation (1) requires approximately $O(N^2)$ complex multiplications. FFT techniques can obtain a reduction to approximately $O(N \log N)$ for certain values of N. For large N, the complexity of FFT techniques is much less than the complexity of direct techniques.

One special property of discrete Fourier transform weights is that they are uniformly spaced on the unit circle in the complex plane. Multiplication of one weight by a unit-amplitude complex number with a phase equal to the uniform phase spacing results in another valid weight. Because of the special properties of the weights, for suitable values of the transform size N, it is possible to decompose a DFT computation into a set of DFT computations of smaller size. Accordingly, it may be possible to decompose each small DFT computation into a set of even smaller DFT computations.

When it is recursively applied, the decomposition leads to a computational structure consisting of successive stages. Each stage consists of multiple small transforms which calculate sums of products. Each small transform has multiple inputs and multiple outputs. Each output of one processing stage is passed to more than one of the small transforms of the next stage. This means that the small transforms of one stage share the calculation results of the prior stage.

The decomposition and implicit shared computation allow FFT techniques to reduce the number of multiplication operations relative to the number required for direct DFT computation. Since the multiplication count is reduced, the cost of implementing

the transform may also be reduced. Subsequent to the decomposition and the resulting reduction in the number of multiplication operations, one may apply the reduced-complexity multiplication techniques discussed above. Constant or non-constant multipliers can replace general multipliers, and so reduce the cost still further.

DESCRIPTION - FIG 1

For instance, the ultimate decomposition of a particular value of N may result in a set of two-point or four-point DFT computations. However, both two-point and four-point DFT computations can have extremely low implementation cost.

Fig 1 shows a structure for computing a 4-point discrete Fourier transform. In the figure there are a DFT input x[0] **10**, a DFT input x[1] **12**, a DFT input x[2] **14**, and a DFT input x[3] **16**. The transform produces a DFT output X[0] **18**, a DFT output X[1] **20**, a DFT output X[2] **22**, and a DFT output X[3] **24**. Each output is the sum of several inputs each weighted by a coefficient. The possible weights for a 4-point DFT are a DFT weight $1 + 0j$ **26**, a DFT weight $0 + j$ **28**, a DFT weight $-1 + 0j$ **30**, and a DFT weight $0 - j$ **32**. In the figure the weight of each input in each output is displayed next to the connecting branch between the two.

Suppose there is an arbitrary complex number with real component A and imaginary component B, for instance, an input of the 4-point DFT in Fig 1.

(2) $(A + Bj)(1 + 0j) = A + Bj$

(3) $(A + Bj)(-1 + 0j) = -A - Bj$

(4) $(A + Bj)(0 + 1j) = -B + Aj$

(5) $(A + Bj)(0 - 1j) = B - Aj$

Equations (2), (3), (4), and (5) show that the products of A+Bj and the four possible weights of the transform in Fig 1 require only negation operations and operations that exchange the real and imaginary components of the arbitrary complex number. Suitable

sets of these operations can replace the general real multipliers that would be used in a general complex multiplier in the transform of Fig 1.

The four-point DFT of Fig 1 can be computed with general multipliers. However, it can have much lower cost when negation and exchange operations replace general multipliers, if those operations have low cost. The potential for reduced cost depends on the special number values of the transform weights and on the low-cost substitute operations.

DESCRIPTION - FFT AND REDUCED-COMPLEXITY MULTIPLIER SUMMARY

FFT techniques can reduce the cost of implementing a DFT or an inverse DFT in two ways. One way is by decomposing a large transform into sets of small transforms. This reduces the number of multiplications required to compute the large transform relative to the number of multiplications required for direct computation. A key feature of this cost reduction is that it depends on the special properties of number values of the DFT or inverse DFT weights. It does not depend on the actual representations of the weights. For instance, the multiplication count savings are the same whether the weights are represented in a 16-bit binary twos complement or in a 24-bit signed integer format.

The second way that FFT techniques can reduce the cost of implementing a DFT or an inverse DFT is by using low-complexity multiplication techniques. These can be existing techniques such as the negation and exchange methods discussed above for Fig 1. They can also be constant multiplier techniques, or non-constant, non-general multiplier techniques. The actual cost of low-complexity multiplication techniques can vary. Key features of the cost reduction can be the number values of the numbers being multiplied, the finite-precision numeric formats in which they are represented, and their actual representations in those formats.

Prior art reduced-complexity multiplication techniques have focused on exploiting number values and representations when computing products one at a time. FFT